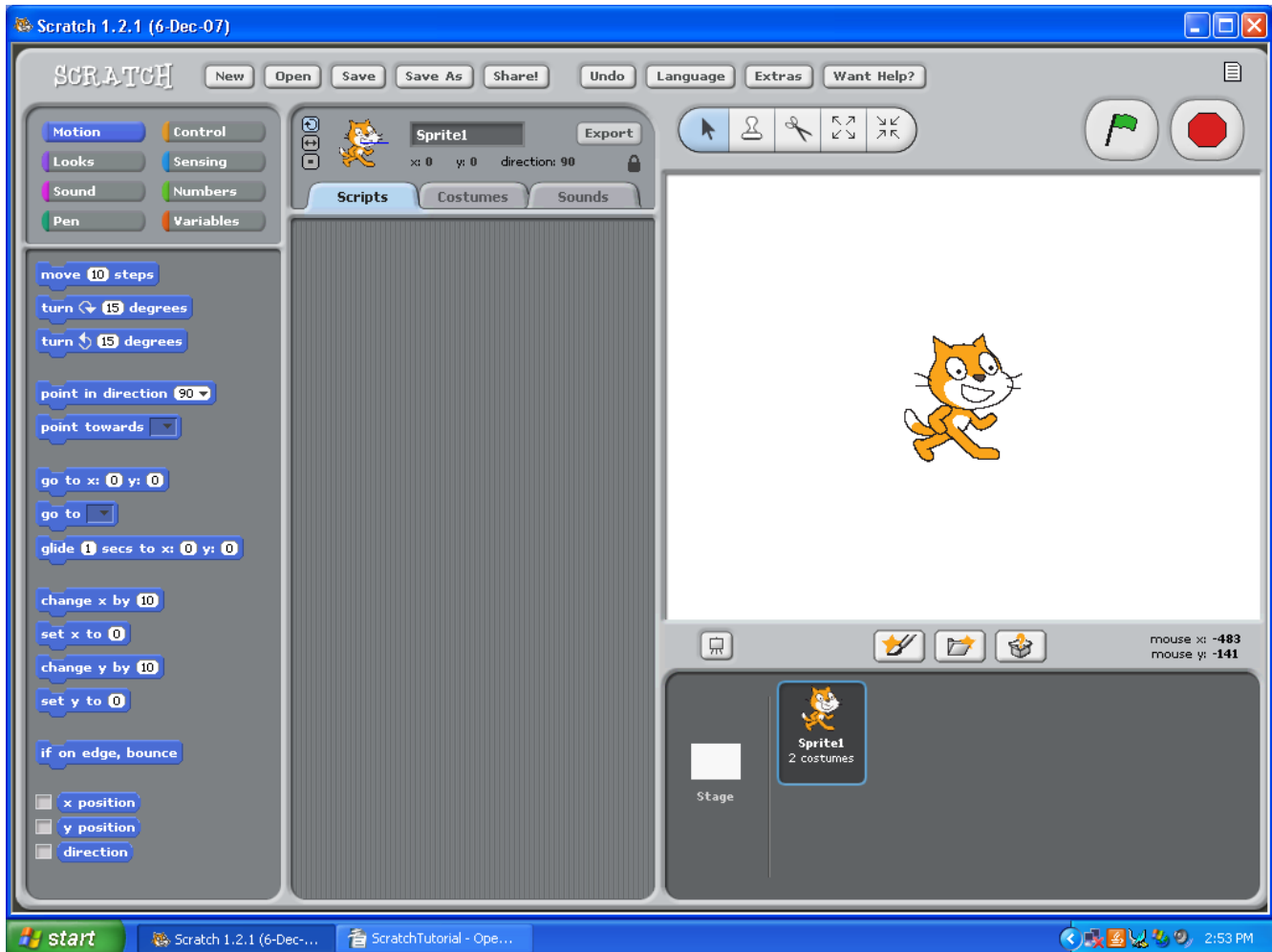


## Maze Game Tutorial

by Ada Taylor

One fun game to make using Scratch is a maze game, because it uses a lot of cool things in it! It also can be as complicated or as easy as you want to make it, and you have plenty of room to show your own creativity. So let's learn some of the basics of a maze game!

Go ahead and open a new Scratch. Once you've opened Scratch, you should see a screen like this:



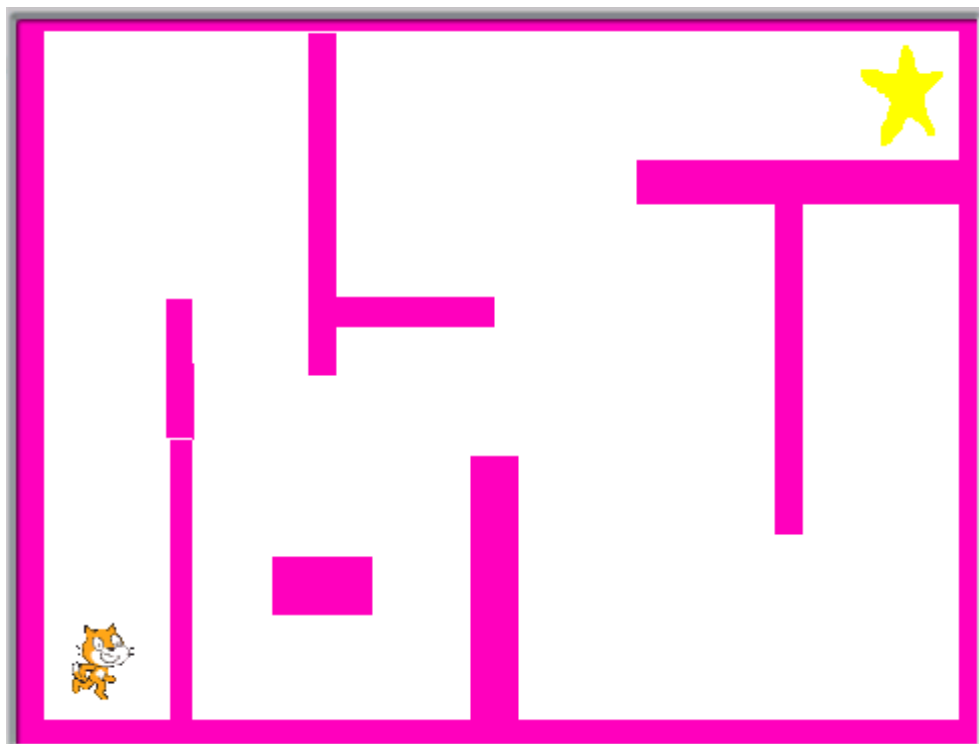
Note: I will almost always be using Windows for my examples. I have used Scratch on Macs before, and there is almost no difference. The only differences that you may find are just locating files, slightly different window management, and other things that are different for all programs on Macs vs Windows. So don't worry. These tutorials should work the same way.

**The first thing that we need to do is to make a maze background or two in the stage area. Make sure that all of the walls of your maze are the same color, because we will be using their color to detect when the person runs into them. Try to make straight walls that are fairly thick. You also want to make sure that there is a way for a sprite to walk through your maze, because otherwise the person playing your game will be very frustrated! You can make the background of your maze any color (or colors) that you want that are not the color of the walls.**


After you have finished making the maze for your sprite, we need to make your cat sprite small enough to move through it. Using the shrinking tool at the top right, shrink the cat until he is small enough to fit through the smallest hall of your maze.

**We also need one more sprite for our maze: a goal sprite! Draw some sort of new sprite to be the target of your maze. It could be something like a pot of gold, a star, or a smiley face. Make sure that it doesn't use any of the color of the wall though, because otherwise the character will ricochet off of the goal instead of reaching it! The goal should be about the size of the player's sprite.**

**Here's what I have:**



**Now for the programming!**

The first thing we want the sprite to do is to move in a direction when we hit one of the arrow keys. To do this, start with a “When  key pressed” starting block.

Underneath it add a motion block that asks the sprite to point in direction “-90 (left)”. After it has pointed in that direction, ask it to move 10 steps. Try out your code!



Oops! You may have noticed that the sprite turned upside-down. This is because Scratch assumes by default that you mean for the sprite to literally change direction. To fix this, go to the top picture of the sprite above the code area. Instead of the arrow going all the way around, pick the arrows pointing back and forth. This will let the sprite only point to the left or the right, and will automatically be flipped right-side up. Hooray!

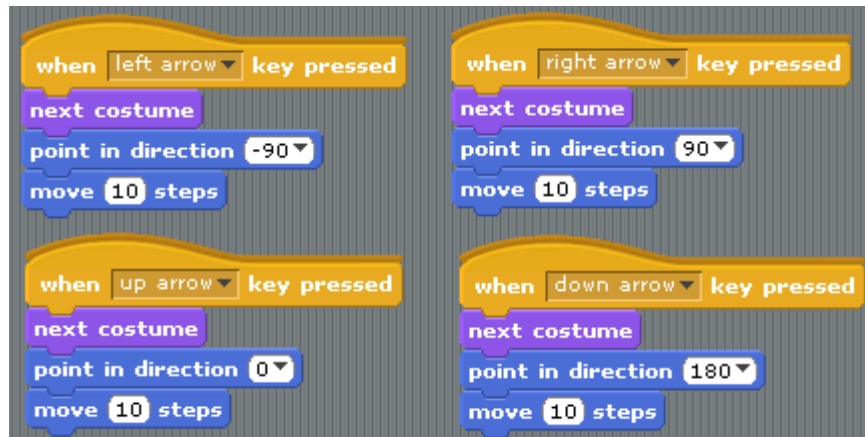


To make the sprite look like it's walking, just put a block that says “next costume” at the top of the stack. Now whenever the sprite walks he will move his legs like he is walking at the same time!



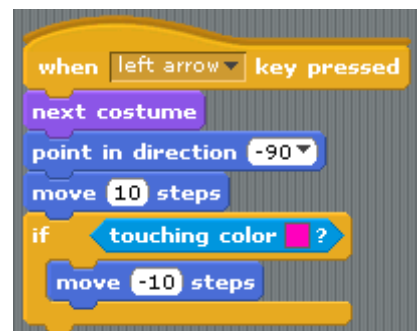
Of course, we want the sprite to be able to move in all directions, not just one. But we don't need to take the time to make a whole new chunk of code for each direction. Instead, we can just copy the entire block three times so that we have four of this block, then change each block so it moves the sprite in a different direction.

To do this, first change all the “when <key> pressed” blocks to each of the different directions, then change all of the “point in direction” blocks to the direction that matches the one at the top of their stack. Now try your program!



Although our sprite can move, you may have noticed that when it collides with walls nothing happens. To change this, we need to have the sprite detect when it collides with a wall, and then bounce off. We can have it do this by asking it to move -10 steps when it is touching the color of the wall. Note that we are not asking the sprite to change its x or y by a certain amount, but to move a negative amount. This means that it will move away from the direction it is currently facing, and that it moved backwards the same amount that it moved forward so it is back in the same spot as before.

You can detect the wall color using a diamond from the sensing area, and pick the exact color of your walls by clicking on the square of color so that an eyedropper comes up and then clicking on part of your wall. The square should change color to match your walls. Add this part of code to all of your movement stacks, so that each one looks something like this:



**You now have a simple maze! Check back for MORE things to add soon!**